UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/007,593	12/05/2001	Roy F. Brabson	RSW920010174US1	3527

7590 07/17/2006

Jerry W. Herndon
IBM Corporation T81/503
PO Box 12195
Research Triangle Park, NC 27709

EXAMINER	
SHAW, YIN CHEN	

ART UNIT	PAPER NUMBER
2135	

DATE MAILED: 07/17/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/007,593	BRABSON ET AL.	
	Examiner	Art Unit	
	Yin-Chen Shaw	2135	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *21 April 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-6,9-23,25 and 26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-6,9-23,25 and 26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This written action is responding to the Request for Continued Examination (RCE) dated on 04/21/2006.
2. Claims 1-6, 9-23, and 25-26 have been submitted for examination.
3. Claims 1-6, 9-23, and 25-26 have been examined and rejected.

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claims 10 and 11 recite limitations that depend on Claim 8. There is insufficient antecedent basis for these limitations since Claim 8 has already been cancelled. For examining purpose, Claims 10 and 11 are to be treated as depending on Claim 1. Appropriate correction is required.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims 1, 18, 21, and 25-26 are rejected under 35 U.S.C. 102(b) as being anticipated by Krause et al. (U.S. Patent 6,070,198).

a. *Referring to Claim 1:*

As per Claim 1, Krause et al. disclose a method of improving security processing in a computing network, comprising:

providing security processing in an operating system kernel **[Kernel space 104 includes stream head 110, encryptor module 112 (lines 38-40, Col. 5 and Fig. 2). Kernel space 124 includes stream head 130, encryption multiplexor 132, TCP layer 134, IP layer 136, DLPI layer 138, and encryptor driver 140 (lines 46-48, Col. 6 and Fig. 3)]**;

providing an application program which makes use of the operating system kernel during execution **[In FIG. 2, when user application 106 desires to write data via a TCP/IP connection, application 106 transmits the data to socket 108 by invoking a system call, such as a send() command. The system call invokes a copyin() function that transfers data from user space 102 into kernel space 104 (lines 41-45, Col. 5 and Fig. 2). FIG. 3 is a diagram of a computer system 120 showing another embodiment of the present invention. Computer system 120 includes user space 122 and kernel space 124. User space 122 includes user application 126 and socket 128. Kernel space 124 includes stream head 130, encryption multiplexor 132, TCP layer 134, IP layer 136, DLPI layer 138, and encryptor driver 140. User application 126, socket 128, stream head 130, TCP layer 134, IP layer 136, and DLPI layer 138 operate in a manner**

similar to correspondingly named components in FIG. 2 (lines 45-51, Col. 6)];

executing the application program **[Execution of the user application 126 (lines 59-60, Col. 7)]**;

selectably securing at least one communication of the executing application program with a remotely executing application program using the provided security processing in the operating system kernel **[This invention relates to data communications within and between computer systems. More particularly, this invention relates to providing data communications within and between computer systems using a modified protocol stack that encrypts and decrypts data as the data flows through the protocol stack (lines 16-21, Col. 1 and Fig. 2 and 3). Prior art cryptography packages, such as the Secured Socket Library (SSL) specification v. 3.0, serve several purposes. Such packages determine the encryption technique to be employed, determine the public and private keys to be used, and route data to and from the encryption technology. The present invention retains these features, and provides several new benefits. For example, a single common interface is used for both network communications and cryptographic technology (lines 29-37, Col. 16). In FIG. 2, when user application 106 desires to write data via a TCP/IP connection (lines 41-42, Col. 5); *where computers***

communicate data for the applications through the network. **Encryptor module 112 encrypts data using software-based algorithms, or hardware-based algorithms, as are known in the art. In addition, user application 106 can selectively add and remove encryption by pushing encryptor module 112 on the stack and pulling encryptor module 112 from the stack (lines 57-61, Col. 5 and Fig. 2). Dynamic function replacement allows a module to have alternate execution paths. By issuing a command, different functions can be switched into and out of a STREAMS-based protocol stack on the fly. Dynamic function replacement is an ideal mechanism for providing a TCP/IP stack with the ability to selectively encrypt and decrypt data flowing through the stack (lines 48-54, Col. 8 and Fig. 3)].**

providing, in the secure processing, support for at least one security directive **[Another mechanism that can be used to differentiate between stacks that encrypt and decrypt at the DLPI layer and those that do not is to specify whether encryption is desired when configuring a network connection using an ifconfig function call. In the Unix operating system, an ifconfig function call is used to assign an address to a network interface and/or configure network interface parameters (lines 26-33, Col. 6). Accordingly, the present invention includes a modified ifconfig function call that is capable**

of defining encryption parameters along with the other network interface parameters (lines 38-41, Col. 6)]; and

invoking, during execution of the provided application program, the at least one security directive **[In FIG. 2, when user application 106 desires to write data via a TCP/IP connection, application 106 transmits the data to socket 108 by invoking a system call, such as a send() command. The system call invokes a copyin() function that transfers data from user space 102 into kernel space 104. Encryptor module 112 encrypts the data, TCP layer 114 and IP layer 116 perform processing in accordance with the TCP/IP protocol, and DLPI layer 118 initiates a DMA operation that provides the data to the physical network hardware (lines 41-50, Col. 5). For example, with the autopush facility properly configured, whenever a user application opens a TCP/IP stack, an encryption module is also pushed onto the stack after TCP layer 114 (lines 2-6, Col. 6). Accordingly, the present invention includes a modified ifconfig function call that is capable of defining encryption parameters along with the other network interface parameters (lines 38-41, Col. 6)]**.

b. *Referring to Claim 18:*

As per Claim 18, Krause et al. disclose the method according claim 1, Krause et al. further disclose wherein the provided application program

comprises security directives that invoke the security processing, and further comprising: Intercepting, in the provided security processing, the security directives **[application 12 must first encrypt the data by sending unencrypted data to encryptor 18 via secured socket 16. Typically, this is accomplished by user application 12 invoking a system call, such as a send() command. The secured socket library associated with secured socket 16 maps the send() command to an implementation-specific write command that provides encryptor 18 with the data to be encrypted. After the data has been encrypted, user application 12 invokes a system call to retrieve the encrypted data, such as a recv() command. The recv() command is mapped to an implementation-specific read command that retrieves the encrypted data from encryptor 18, and the encrypted data is then provided to user application 12. Thereafter, user application 12 transmits the encrypted data over the network by sending the encrypted data to stream head 22 via socket 14 by invoking another system call (such as another send() command). From stream head 22, the encrypted data flows through TCP layer 24, IP layer 26, and DLPI layer 28 (lines 66-67, Col. 2 and lines 1-18, Col. 3). In FIG. 2, when user application 106 desires to write data via a TCP/IP connection, application 106 transmits the data to socket 108 by invoking a system call, such as a send() command.**

The system call invokes a copyin() function that transfers data from user space 102 into kernel space 104. Encryptor module 112 encrypts the data (lines 41-46, Col. 5 and Fig. 2). Another mechanism that can be used to differentiate between stacks that encrypt and decrypt at the DLPI layer and those that do not is to specify whether encryption is desired when configuring a network connection using an ifconfig function call. In the Unix operating system, an ifconfig function call is used to assign an address to a network interface and/or configure network interface parameters (lines 26-33, Col. 6). Accordingly, the present invention includes a modified ifconfig function call that is capable of defining encryption parameters along with the other network interface parameters (lines 38-41, Col. 6)]; and

executing, responsive to the interception, corresponding security function **[Encryptor module 112 encrypts the data using software-based algorithms, or hardware-based algorithms (lines 57-58, Col. 5). Dynamic function replacement allows a module to have alternate execution paths. By issuing a command, different functions can be switched into and out of a STREAMS-based protocol stack on the fly. Dynamic function replacement is an ideal mechanism for providing a TCP/IP stack with the ability to**

selectively encrypt and decrypt data flowing through the stack (lines 48-54, Col. 8 and Fig. 3)].

c. *Referring to Claim 21:*

As per Claim 21, Krause et al. discloses the method according to claim 1. Krause et al. further disclose the provided security processing operaties in a Transmission Control Protocol layer of the operating system kernel **[In addition, compute 100 can be configured to add encryption to any TCP/IP stack using the STREAMS autopush facility. The autopush facility allows modules to be automatically pushed on the stack by the operating system whenever predefined types of stacks are open (lines 65-67, Col. 5, lines 1-2, Col. 6, and Fig. 2 and 3)]**.

d. *Referring to Claim 25:*

As per Claim 25, it encompasses limitations that are similar to those of the method Claim 1. Therefore, it is rejected with the same rationale applied against Claim 1 above. In additional, Krause et al. disclose a system for improving security processing in a computing network **[This invention relates to data communications within and between computer systems. More particularly, this invention relates to providing data communications within and between computer systems using a modified protocol stack that encrypts and decrypts data as the data flows through the protocol stack (lines**

16-21, Col. 1). A single common interface is used for both network communications and cryptographic technology, thereby simplifying user applications (lines 35-38, Col. 16)].

e. *Referring to Claim 26:*

As per Claim 26, it encompasses limitations that are similar to those of the method Claim 1. Therefore, it is rejected with the same rationale applied against Claim 1 above. In additional, Krause et al. disclose a computer program product for improving security processing in a computing network, the computer program product embodied on at least one computer-readable media **[A program storage medium readable by a computer tangibly embodying a program of instructions executable by the computer to perform a cryptographic function (lines 7-9, Col. 21)]**.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 2-6, 9-17, 19, and 22-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Krause et al. (U.S. Patent 6,070,198) and further in view of Mod_SSL manual (Apache mod_ssl version 2.6).

   a. *Referring to Claim 2:*

      As per Claim 2, Krause et al. disclose the method according to claim 1. In addition, Krause et al. further disclose the step of configuring at least one port used by the provided application program such that communications using the at least one port are to be secured **[The replacement put function can also be constructed to scan for TPI T_BIND_REQ messages. TPI T_BIND_REQ messages are used to bind an application to a specified protocol stack and port. In a manner similar to that described above, the replacement put function can determine what level of encryption is required, determine the encryption keys, and register the encryption function at the stream head (lines 4-10, Col. 14). For example, a single common interface is used for both network communications and cryptographic technology, thereby simplifying user applications (lines 35-38, Col. 16)]**; and

      wherein selectably securing the at least one communication of the executing application program and using the at least one port **[Encryptor module 112 encrypts data using software-based algorithms, or hardware-based algorithms, as are known in the art.**

In addition, user application 106 can selectively add and remove encryption by pushing encryptor module 112 on the stack and pulling encryptor module 112 from the stack (lines 57-61, Col. 5 and Fig. 2). Dynamic function replacement allows a module to have alternate execution paths. By issuing a command, different functions can be switched into and out of a STREAMS-based protocol stack on the fly. Dynamic function replacement is an ideal mechanism for providing a TCP/IP stack with the ability to selectively encrypt and decrypt data flowing through the stack (lines 48-54, Col. 8 and Fig. 3). The replacement put function can also be constructed to scan for TPI T_BIND_REQ messages. TPI T_BIND_REQ messages are used to bind an application to a specified protocol stack and port. In a manner similar to that described above, the replacement put function can determine what level of encryption is required, determine the encryption keys, and register the encryption function at the stream head (lines 4-10, Col. 14)].

Krause et al. discloses selectably securing the at elast one communication as in Claim 1.

Krause et al. do not expressly disclose wherein selectably securing then secures all communications. Mod_SSL manual discloses selectably securing can secure all communications when the security engine is

turned on **[i.e., SSLEngine on (line 19, pg. 6 of Chapter 3). This directive toggles the usage of the SSL/TLS Protocol Engine. This is usually used inside a <VirtualHost> section to enable SSL/TLS for a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for both the main server and all configured virtual hosts (lines 26-28, pg. 6 of Chapter 3),** ***the security function, SSLEngine, would be applied to all the application or data once it is toggled on*****]**. Krause et al. and Mod_SSL manual are from similar technology relating to network security communications. It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine Krause et al. and Mod_SSL manual since one would be motivated to have SSL Engine configured for securing the network communication between client and server (line 37, pg. 5 of Chapter 2 from Mod_SSL module) all the time and support backward compatibility to other SSL solutions (line 5, pg. 1 of Chapter 4 from Mod_SSL manual). Therefore, it would have been obvious to combine Krause et al. with Mod_SSL manual to obtain the invention as specified in claim 2.

b. *Referring to Claim 3:*

As per Claim 3, Krause et al. and Mod_SSL manual disclose the method according to claim 2. In addition, Krause et al. disclose wherein the provided application program does not include code for security processing **[In FIG. 2, when user application 106 desires to write**

data via a TCP/IP connection, application 106 transmits the data to socket 108 by invoking a system call, such as a send() command. The system call invokes a copyin() function that transfers data from user space 102 into kernel space 104. Encryptor 112 encrypts the data (lines 41-46, Col. 5 and Fig. 2). Encryptor module 112 encrypts data using software-based algorithms, or hardware-based algorithms, as are known in the art (lines 57-59, Col. 5 and Fig. 2). FIG. 3 is a diagram of a computer system 120 showing another embodiment of the present invention. Computer system 120 includes user space 122 and kernel space 124. User space 122 includes user application 126 and socket 128. Kernel space 124 includes stream head 130, encryption multiplexor 132, TCP layer 134, IP layer 136, DLPI layer 138, and encryptor driver 140. User application 126, socket 128, stream head 130, TCP layer 134, IP layer 136, and DLPI layer 138 operate in a manner similar to correspondingly named components in FIG. 2 (lines 45-51, Col. 6 and Fig. 3). By using the present invention, data can be encrypted using keys provided by the kernel, in contrast to the application (lines 28-30, Col. 4). In addition, the user application does not have to support encryption (lines 32-33, Col. 4); *the application program pass the data to the kernel space for security process since it does not provide the security process*].

c. *Referring to Claim 4:*

As per Claim 4, Krause et al. and Mod_SSL manual disclose the method according to claim 2. In addition, Krause et al. disclose configuring at least one port further comprises specifying information to be used in selectably securing the at least one communication of the executing application program **[Another mechanism that can be used to differentiate between stacks that encrypt and decrypt at the DLPI layer and those that do not is to specify whether encryption is desired when configuring a network connection using an ifconfig function call (lines 26-30, Col. 6). Accordingly, the present invention includes a modified ifconfig function call that is capable of defining encryption parameters along with the other network interface parameters (lines 37-40, Col. 6). The encryption key may be bound to the application instance, such that all data originating from a specific application uses a specific key (lines 25-28, Col. 7). the Secured Socket Library (SSL) specification v. 3.0 defined a protocol that allows a cryptographic package to pass messages between the package and the user application to configure how encryption and decryption will proceed, including the encryption keys that will be used, the connections that will be encrypted, and the like (lines 66-67, Col. 6 and lines 1-4, Col. 7)]**.

d. *Referring to Claim 5:*

As per Claim 5, Krause et al. and Mod_SSL manual disclose the method according to claim 4. In addition, Mod_SSL manual disclose wherein the specified information comprises at least one of: authentication information; cipher suites options; and security key input information **[i.e., This complex directive uses a colon-separated cipher-spec string consisting of OpenSSL cipher specifications to configure Cipher Suit the client is permitted to negotiate in the SSL handshake phase (line 1-2, pg. 8 of Chapter 3)]**.

e. *Referring to Claim 6:*

As per Claim 6, Krause et al. and Mod_SSL manual disclose the method according to claim 2. In addition, Mod_SSL manual disclose wherein configuring at least one port comprises at least one of: providing port definition statements; setting environment variables; and using job control language **[i.e., this module provides a lot of SSL information as additional environment variable to the SSI and CGI namespace. For backward compatibility the information can be made available under different names, too (lines 30-32, pg. 20 of Chapter 3)]**.

f. *Referring to Claim 9:*

As per Claim 9, Krause et al. disclose the method according to claim 1. Krause et al. disclose the at least one security directive as in Claim 1. Krause et al. do not expressly disclose the at least one security directive comprises at least one: access capability for a client certificate; access

capability for a client identifier; a request to start operation of selectably securing the at least one communication of the executing application program; and a request to stop operation of the selectably securing the at least one communication of the executing application program. However, Mod_SSL manual further discloses wherein the at least one security directive comprises at least one:

access capability for a client certificate; access capability for a client identifier; a request to start operation of selectably securing the at least one communication of the executing application program; and a request to stop operation of the selectably securing the at least one communication of the executing application program **[i.e., SSLEngine (line 16, pg. 6 of Chapter 3). This directive toggles the usage of the SSL/TLS Protocol Engine. This is usually used inside a <VirtualHost> section to enable SSL/TLS for a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for both the main server and all configured virtual hosts (lines 26-28, pg. 6 of Chapter 3);** ***where the SSL/TLS engine is used to provide the security process for the network communication involving the application data in pg. 9, Chapter 2 of the Mod_SSL manual***].

Krause et al. and Mod_SSL manual are from similar technology relating to network security communications. It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine

Krause et al. and Mod_SSL manual to have various types of directives being supported as the security processing for the application since one would be motivated to know how a particular mod_ssl functionality is actually configured or activated (lines 3-4, pg. 1 of Chapter 3 from Mod_SSL manual). Therefore, it would have been obvious to combine Krause et al. and Mod_SSL manual to obtain the invention as specified in claim 9.

g. *Referring to Claim 10:*

As per Claim 10, Krause et al. and Mod_SSL manual disclose the method according to claim 1. Krause et al. further disclose the step of invoking the at least one of the security directive and the executing application program as in Claim 1. Krause et al. do not expressly disclose the security directive comprises an access capability, a client certificate, and returning the client certification from the provided security processing in response to the invocation. However, Mod_SSL manual discloses the security directive comprises an access capability, a client certificate, and returning the client certification from the provided security processing in response to the invocation **[i.e., SSLVerifyClient (line 1, pg. 14 of Chapter 3). This directive sets the Certificate verification level for the Client Authentication. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the**

standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotation with the reconfigured client verification level after the HTTP request was read but before the HTTP response is sent (lines 11-15, pg. 14 of Chapter 3); *the process of authentication means that the certificate is required to be accessed and then returned for verification*].

Krause et al. and Mod_SSL manual are from similar technology relating to network security communications. It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine Krause et al. and Mod_SSL manual to have various types of directives being supported as the security processing for the application since one would be motivated to know how a particular mod_ssl functionality is actually configured or activated (lines 3-4, pg. 1 of Chapter 3 from Mod_SSL manual) and have the have the client authentication for the standard SSL handshake (lines 12-13, pg. 14 of Chapter 3). Therefore, it would have been obvious to combine Krause et al. and Mod_SSL manual to obtain the invention as specified in claim 10.

h. *Referring to Claim 11:*

As per Claim 11, it encompasses limitations that are similar to those of method Claims 10. Therefore, it is rejected with the same rationale applied against Claim 11 above. In addition, Mod_SSL manual discloses the client identification **[i.e., Certificate (line 11, pg. 14 of**

Chapter 3), *where the certificate would contain a distinguished names as defined by the X.509 standard, and the distinguished name is used to provide an identity in a specific context (Table 2, pg. 3-4 in Chapter 2)*].

i. *Referring to Claim 12:*

As per Claim 12, the rejection of Claim 1 is incorporated. Claim 12 further encompasses limitations that are similar to those of method Claims 1, 2, and 9. Therefore, it is rejected with the same rationale as applied to Claims 1, 2, and 9. In addition, Mod_SSL manual discloses wherein selectably securing then secures all communications **[i.e., SSLEngine on (line 19, pg. 6 of Chapter 3). This directive toggles the usage of the SSL/TLS Protocol Engine. This is usually used inside a <VirtualHost> section to enable SSL/TLS for a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for both the main server and all configured virtual hosts (lines 26-28, pg. 6 of Chapter 3), *the security function, SSLEngine, would be applied to all the application or data once it is toggled on*]**.

j. *Referring to Claim 13:*

As per Claim 13, the rejection of Claim 1 is incorporated. Claim 13 further encompasses limitations that are similar to those of method Claims 1 and 9. Therefore, it is rejected with the same rationale as applied to Claims 1 and 9. In addition, Mod_SSL manual discloses

wherein selectably securing the at least one communication of the executing application program then comprises stopping securing communications of the executing application program **[i.e., SSLEngine off (line 19, pg. 6 of Chapter 3). This directive toggles the usage of the SSL/TLS Protocol Engine. This is usually used inside a <VirtualHost> section to enable SSL/TLS for a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for both the main server and all configured virtual hosts (lines 26-28, pg. 6 of Chapter 3 and pg. 9, Chapter 2 of the Mod_SSL), *the security function, SSLEngine, would no longer be applied to all the data for application in the network communication when it is toggled off*]**.

k. *Referring to Claim 14:*

As per Claim 14, the rejection of Claim 12 is incorporated. In addition, Claim 14 encompasses limitations that are similar to those of the method Claim 4. Therefore, it is rejected with the same rationale applied against Claim 4 above.

l. *Referring to Claim 15:*

As per Claim 15, the rejection of Claim 14 is incorporated. In addition, Claim 15 encompasses limitations that are similar to those of the method Claim 5. Therefore, it is rejected with the same rationale applied against Claim 5 above.

m. *Referring to Claim 16:*

As per Claim 16, Krause et al. and Mod_SSL manual disclose the method according to claim 12. Krause et al. further disclose a decision to invoke and the executing application program **[In FIG. 2, when user application 106 desires to write data via a TCP/IP connection, application 106 transmits the data to socket 108 by invoking a system call, such as a send() command. The system call invokes a copyin() function that transfers data from user space 102 into kernel space 104. Encryptor module 112 encrypts the data (lines 41-45, Col. 5 and Fig. 2). FIG. 3 is a diagram of a computer system 120 showing another embodiment of the present invention. Computer system 120 includes user space 122 and kernel space 124. User space 122 includes user application 126 and socket 128. Kernel space 124 includes stream head 130, encryption multiplexor 132, TCP layer 134, IP layer 136, DLPI layer 138, and encryptor driver 140. User application 126, socket 128, stream head 130, TCP layer 134, IP layer 136, and DLPI layer 138 operate in a manner similar to correspondingly named components in FIG. 2 (lines 45-51, Col. 6); *where the application program is executed and call to the encryptor at kernel for required data security*]**.

n. *Referring to Claim 17:*

As per Claim 17, the rejection of Claim 12 is incorporated. Claim 17 encompasses limitations that are similar to those of the method Claim

16. Therefore, it is rejected with the same rationale applied against Claim 16 above. In addition, Mod_SSL manual discloses a security negotiation protocol **[i.e., The protocol is designed to support a range of choices for specific algorithms used for cryptography, digests, and signatures. This allows algorithm selection for specific servers to be made based on legal, export or other concerns, and also enables the protocol to take advantage of new algorithms. Choices are negotiated between client and server at the start of establishing a protocol session. (lines 38-42, pg. 5 of Chapter 2). SSLCipherSuite (line 29, pg. 7 of Chapter 3), where Cipher Suite available for negotiation in SSL handshake (line 30, pg. 7 of Chpater 3)]**.

o. *Referring to Claim 19:*

As per Claim 19, Krause et al. disclose the method according to claim 1. Krause et al. further disclose wherein the provided application program comprises security directives that invoke the security processing **[application 12 must first encrypt the data by sending unencrypted data to encryptor 18 via secured socket 16. Typically, this is accomplished by user application 12 invoking a system call, such as a send() command. The secured socket library associated with secured socket 16 maps the send() command to an implementation-specific write command that provides encryptor 18**

with the data to be encrypted. After the data has been encrypted, user application 12 invokes a system call to retrieve the encrypted data, such as a recv() command. The recv() command is mapped to an implementation-specific read command that retrieves the encrypted data from encryptor 18, and the encrypted data is then provided to user application 12. Thereafter, user application 12 transmits the encrypted data over the network by sending the encrypted data to stream head 22 via socket 14 by invoking another system call (such as another send() command). From stream head 22, the encrypted data flows through TCP layer 24, IP layer 26, and DLPI layer 28 (lines 66-67, Col. 2 and lines 1-18, Col. 3). In FIG. 2, when user application 106 desires to write data via a TCP/IP connection, application 106 transmits the data to socket 108 by invoking a system call, such as a send() command. The system call invokes a copyin() function that transfers data from user space 102 into kernel space 104. Encryptor module 112 encrypts the data (lines 41-46, Col. 5 and Fig. 2). Another mechanism that can be used to differentiate between stacks that encrypt and decrypt at the DLPI layer and those that do not is to specify whether encryption is desired when configuring a network connection using an ifconfig function call. In the Unix operating system, an ifconfig function call is used to assign an address to a network interface and/or

configure network interface parameters (lines 26-33, Col. 6). Accordingly, the present invention includes a modified ifconfig function call that is capable of defining encryption parameters along with the other network interface parameters (lines 38-41, Col. 6)]. Krause et al. do not expressly disclose interpreting, in the provided security processing, the security directives as being non-operative. However, the Mod_SSL manual discloses the SSLRequire directive, which only allows the access when certain boolean condition is true **[i.e., SSLRequire (line 14, pg. 18 of Chapter 3) allow access only when an arbitrarily complex boolean expression is true (line 15, pg. 18 of Chapter 3); *This means that under certain conditions (depending on the setup of the expressions), the SSLRequire directive will be treated as disabled (non-operative)*]**. Krause et al. and Mod_SSL manual are from similar technology relating to network security communications. It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine Krause et al. and Mod_SSL manual to determine Boolean condition since one would be motivated to have directive that specifies a general access requirement which has to be fulfilled in order to allow access (lines 23, pg. 18 of Chapter 3 from Mod_SSL manual), which functions similarly to the process of interpreting whether the call is operative or non-operative.

Therefore, it would have been obvious to combine Krause et al. and Mod_SSL manual to obtain the invention as specified in claim 19.

p. *Referring to Claim 22:*

As per Claim 22, Krause et al. disclose the method according to claim 1. Krause et al. do not expressly disclose wherein the provided security processing implements Secure Socket Layer. However, Mod_SSL manual discloses the module can provide security as secure socket layer **[i.e., This module provides strong cryptography for the Apache (v1.3) webserver via the Secure Socket Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols (lines 1-3, pg. 1 of Chapter 1)]**. Krause et al. and Mod_SSL manual are from similar technology relating to network security communications. It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine Krause et al. and Mod_SSL manual to realize the security process can either be implemented as SSL or TLS since one would be motivated to establish the server environment such that the clients can only connect with one of the provided protocols (lines 8-9, pg. 7 of Chapter 3 from Mod_SSL manual). Therefore, it would have been obvious to combine Krause et al. and Mod_SSL manual to obtain the invention as specified in claim 22.

q. *Referring to Claim 23:*

As per Claim 23, Krause et al. disclose the method according to claim 1. Krause et al. do not expressly disclose wherein the provided security processing implements Transport Layer Security. However, Mod_SSL manual discloses the module can provide security as transport layer security **[i.e., This module provides strong cryptography for the Apache (v1.3) webserver via the Secure Socket Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols (lines 1-3, pg. 1 of Chapter 1)]**. Krause et al. and Mod_SSL manual are from similar technology relating to network security communications. It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine Krause et al. and Mod_SSL manual to realize the security process can either be implemented as SSL or TLS since one would be motivated to establish the server environment such that the clients can only connect with one of the provided protocols (lines 8-9, pg. 7 of Chapter 3 from Mod_SSL manual). Therefore, it would have been obvious to combine Krause et al. and Mod_SSL manual to obtain the invention as specified in claim 23.

7. Claim 20 is rejected under 35 U.S.C. 103(a) as being unpatentable over Krause et al. (U.S. Patent 6,070,198) and further in view of Smith et al. (U.S. Patent 6,801,927).

   a. *Referring to Claim 20:*

As per Claim 20, Krause et al. disclose the method according to claim 18. Krause et al. and Golan do not expressly disclose wherein the provided application program may be executed on a system which does not include the provided security processing in the operating system kernel, in which case the calls operate to perform security processing instead of selectably securing the at least one communication of the executing application program. However, Smith et al. disclose the network adapter card, connected to the server computer, with security processing capability stored in the memory of the network adapter card, rather than in the kernel of the operating system, for off-loading the CPU task while performing the security to the communicating application **[Server 106 includes non-volatile memory 110, working memory 112, server mass data storage 114, a processing unit 116, and one or more user input/output (I/O) devices 118, all intercommunicating via a server bus 120 (e.g., PCI bus). Non-volatile memory 110 (e.g., read-only memory and/or one or more hard-disk drives) provides storage for data and code which is retained even when server 106 is powered down. Working memory 112 (e.g., random access memory) provides operational memory for server 106, and includes executable code (e.g., an operating system) which is loaded into working memory 112 during start-up (lines 6-16, Col. 5). The invention facilitates off-loading the connection management**

burden from the host CPU to an adapter card interposed between the network and the host bus (lines 62-64, Col. 1). FIG. 3 is a block diagram showing application proxies module 208 to include a plurality of application specific proxies 208(1-f), including a hypertext transfer protocol (HTTP) proxy 208(1), a pass-through proxy 208(2), a security proxy 208(3), a caching proxy 208(4), and an "other" proxy 208(f) (lines 16-21, Col 7 and Fig. 1, 2, and 3); ***where the security proxy performs the security processing for the application when the security system does not have security processing capability in the kernel space of the operating system***]. Krause et al. and Smith et al. are from similar technology relating to network security communications. It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine Krause et al. with Smith et al. to have the security processing off-loaded to adapter card when it is not available in the kernel space since one would be motivated to off-load the connection management burden from the host CPI to an adapter card interposed between the network and the host bus (lines 62-64, Col. 1 from Smith et al.). Therefore, it would have been obvious to combine Krause et al. with Smith et al. to obtain the invention as specified in claim 20.

Conclusion

8. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

   a. Carter (U.S. Patent 7,000,106) discloses a computer-readable medium containing program instructions for configuring a first computer so that a first telephony client on the first computer may securely communicate with a second telephony client on a second computer via a communication path. The computer-readable medium includes computer code for inserting a security algorithm within the communication path. The security algorithm facilitates secure communication between the first and second telephony clients such that more than a single type of telephony client may be implemented. In a specific embodiment, the security algorithm is inserted within the first computer's operating system kernel. Carter further discloses the security setting specified by the user and the security flag to indicate whether encryption/decryption is set.

   b. Choo (U.S. Patent 6,981,140) discloses a kernel/operating system from faults arising from the operation of the internet protocol security stack during encryption or decryption of user data an internet protocol security stack is located within the second area of memory 503 which is logically distinct from the first memory area 502 containing the kernel/operating system. The second memory area 503 comprises a plurality of locations within a total memory means of the computer which are allocated for

storing user applications programs. Fig. 5 illustrates that the operating system comprises a set of code referred to as a `kernel` which performs the basic functions of input and output functions including communications, as well as memory management functions. In addition to the kernel code, the operating system comprises a non-kernel code for performing other functions of the operating system, for example functions for viewing directories or copying files. The operating system code comprises code for operating network protocols, according to a protocol stack including protocols for the transmission control protocol/internet protocol (TCP/IP)

9. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Yin-Chen Shaw whose telephone number is 571-272-8593. The examiner can normally be reached on 8:00 to 4:00 M-F.

   If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kim Yen Vu can be reached on 571-272-3859. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

   Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR

only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

YCS

Jul. 07, 2006

HOSUK SONG
PRIMARY EXAMINER